

**This Page Is Inserted by IFW Operations
and is not a part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- **BLACK BORDERS**
- **TEXT CUT OFF AT TOP, BOTTOM OR SIDES**
- **FADED TEXT**
- **ILLEGIBLE TEXT**
- **SKEWED/SLANTED IMAGES**
- **COLORED PHOTOS**
- **BLACK OR VERY BLACK AND WHITE DARK PHOTOS**
- **GRAY SCALE DOCUMENTS**

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平6-266562

(43)公開日 平成 6 年(1994) 9 月22日

(51)Int.Cl.⁵

G 0 6 F 9/44
9/45

識別記号

3 3 0 Z

庁内整理番号

9193-5B

9292-5B

F I

G 0 6 F 9/ 44

技術表示箇所

3 2 2 F

審査請求 有 請求項の数 3 F D (全 8 頁)

(21)出願番号 特願平5-80164

(22)出願日 平成 5 年(1993) 3 月15日

(71)出願人 000004237

日本電気株式会社

東京都港区芝五丁目 7 番 1 号

(72)発明者 宮本 敬士

東京都港区芝五丁目 7 番 1 号 日本電気株
式会社内

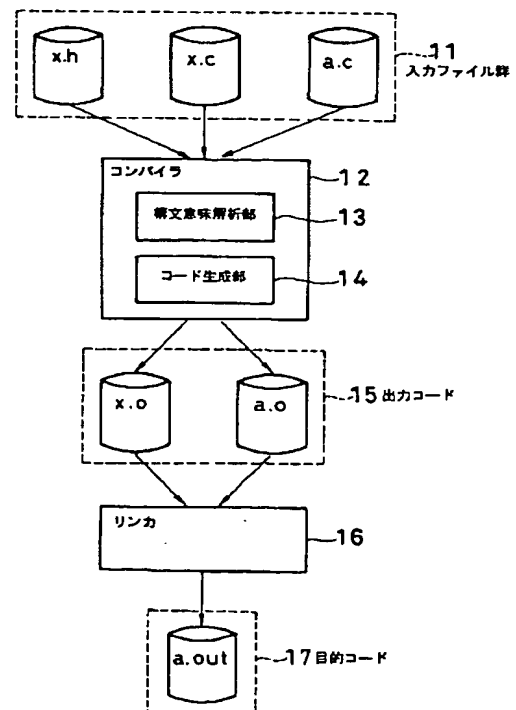
(74)代理人 弁理士 河原 純一

(54)【発明の名称】 オブジェクト指向言語処理システムにおける目的コードサイズ最適化方式

(57)【要約】

【目的】 目的コード内にメソッドアドレス格納テーブルの実体を1つのみ生成し、目的コードの大きさを削減する。

【構成】 入力ファイル群 11 の各入力ファイルに対して出力コード 15 を生成し、かつオブジェクト指向言語のメソッドの実行時決定を目的コード 17 内に静的に生成されるメソッドアドレス格納テーブルを使用することによって実現するオブジェクト指向言語処理システムにおいて、構文意味解析部 13 は、クラス宣言の意味解析処理時にメソッドの所属するクラス名と入力ファイル名とを比較し、一致すればメソッドアドレス格納テーブルの実体および宣言を出力コード 15 内に生成し、一致しなければメソッドアドレス格納テーブルの宣言のみを出力コード 15 内に生成する。リンカ 16 は、複数の出力コード 15 間の参照関係を解決して1つの目的コード 17 を生成する。



【特許請求の範囲】

【請求項 1】 複数の入力ファイルに対して各々に対応する出力コードを生成した後に複数の出力コード間の参照関係を解決して 1 つの目的コードを生成し、かつオブジェクト指向言語のメソッドの実行時決定を目的コード内に静的に生成されるメソッドアドレス格納テーブルを使用することによって実現するオブジェクト指向言語処理システムにおいて、

クラス宣言の意味解析処理時にメソッドの所属するクラス名と入力ファイル名とを比較し、一致すればメソッドアドレス格納テーブルの実体および宣言を出力コード内に生成し、一致しなければメソッドアドレス格納テーブルの宣言のみを出力コード内に生成する構文意味解析部を設けたことを特徴とするオブジェクト指向言語処理システムにおける目的コードサイズ最適化方式。

【請求項 2】 前記クラス名と入力ファイル名とが一致しない場合は、オブジェクト指向言語処理システムが処理対象としているすべての入力ファイル名を調べ、入力ファイル名がクラス名と一致する入力ファイルがあればその入力ファイルの内容を走査し、現在対象としているクラスの宣言が含まれていなければその入力ファイル内にクラスの宣言を含め、入力ファイル名がクラス名と一致する入力ファイルがなければクラス名と同名の入力ファイルを作成し、この入力ファイルにクラスの宣言を含めるとともにこの入力ファイルを入力ファイル群に含める請求項 1 記載の目的コードサイズ最適化方式。

【請求項 3】 前記オブジェクト指向言語が C++ であり、前記メソッドアドレス格納テーブルが仮想関数テーブルでなる請求項 1 または 2 記載のオブジェクト指向言語処理システムにおける目的コードサイズ最適化方式。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明はオブジェクト指向言語（例えば、プログラミング言語 C++、LOOPS、TAO 等）を用いた情報処理システムに関し、特に複数の入力ファイルに対して各々に対応する出力コードを生成した後に複数の出力コード間の参照関係を解決して 1 つの目的コードを生成し、かつオブジェクト指向言語のメソッドの実行時決定を目的コード内に静的に生成されるメソッドアドレス格納テーブルを使用することによって実現するオブジェクト指向言語処理システムにおける目的コードサイズ最適化方式に関する。

【0002】

【従来の技術】 オブジェクト指向言語にはメソッドの実行時決定という機能がある。これは異なるクラスに属するオブジェクトを統括して扱うために、オブジェクトに与えられたメッセージに対して起動されるメソッドをオブジェクトの所属するクラスに基づいて実行時に決定する機能である。

【0003】 ある種のオブジェクト指向言語処理システ

ムでは、この機能を目的コード内に静的に生成されるメソッドアドレス格納テーブルを使用することによって実現している。実際に実行されるメソッドはオブジェクトの所属するクラスに基づいて決定されるから、メソッドアドレス格納テーブルの内容はクラスごとに（正確にはクラスとクラスの継承関係ごとに）異なるものである。

【0004】 従来は、実行時に決定されるメソッドの属するクラスの宣言が入力ソースプログラムファイル（以下、単に入力ファイルという）に含まれていると、オブジェクト指向言語処理システムのユーザが特に指定しない場合は出力コード内にメソッドアドレス格納テーブルの実体が必ず生成されていた。

【0005】 以下、オブジェクト指向言語の 1 つである C++ の場合について従来技術を説明する。C++ には、仮想関数というメソッドの実行時決定のための機構がある。仮想関数であることを指定するためには、virtual というキーワードを用いる。

【0006】 仮想関数を実現するためには、与えられたメッセージ呼出しに対して、実行時（プログラム動作時）に呼び出されるメソッドを決定する必要がある。仮想関数の呼出しが正しく行われるためには、実行されるメソッドが各オブジェクトのもつ情報をもとにプログラム動作時に決定されなければならない。

【0007】 これを実現する方法として自然に考えられる方法は、C++ 処理系がクラスの属性を表す情報を出力コード中に埋め込み、各オブジェクトはその情報へのポインタを保持するという方法である。当然のことながら、クラスの属性を表す情報には実行されるメソッドの情報も含まれている。プログラム動作時には、オブジェクトからポインタをたどってこの情報をアクセスし、実行されるべきメソッドを決定することになる。

【0008】 仮想関数テーブルは、C++ 処理系が仮想関数機能を実現するために出力コード中に埋め込むメソッドアドレス格納テーブルである。仮想関数テーブルは、「クラスの属性を表す情報」を簡略化したものと考えてよい。すなわち、クラスに関する全ての情報をもつのではなく、実行されるべきメソッドの決定を行うために必要最低限の情報をもっているテーブルである。ここでは、仮想関数テーブルは、実行されるべきメソッドのアドレス（位置）を情報としてもっていると考えてよい。

【0009】 仮想関数テーブルが必要なのは、仮想関数（virtual 付きの関数）がクラス宣言中に用いられた場合である。仮想関数が書き換えられるごとに、言い換えれば仮想関数が書き換えられるようなクラスが宣言されるごとに仮想関数テーブルを生成しなければならない。

【0010】 また、入力ファイル中でオブジェクトを使用するには、そのオブジェクトが所属するクラスの宣言が入力ファイル中に含まれていなければならない。ある

クラスのオブジェクトを複数の入力ファイルで使用する
場合、複数の入力ファイルにクラスの宣言が含まれてい
なければならない。このために、通常、クラスの宣言を
ヘッダファイル（ファイル名末尾を、hとすることが多い）
に入れ、オブジェクトを使用する入力ファイルでヘッ
ダファイルをインクルードする（C++の言語仕様
で、`#include`を用いて指定する）ことが多い。
ヘッダファイルのインクルードを指定すると、C++処
理系は、`#include`が指定された部分をインクル
ードされるヘッダファイルの内容で置き換える。

【0011】C++処理系が出力コード中に仮想関数テ
ーブルを埋め込むとき、仮想関数テーブルは、ソースプ
ログラム中の外部変数と同じ形態で埋め込まれるが、外
部変数と同じ形態で埋め込まれるためには、外部変数と
同様に、仮想関数テーブルに名前（変数名と同様）が与
えられる必要がある。このように名前が与えられるか
ら、オブジェクトが仮想関数テーブルへのポインタを保
持することが可能なのである（名前を使って参照するこ
とができる）。逆にいうと、仮想関数テーブルの名前さ
えわかっていれば、仮想関数テーブルの実体（先に述べ
た外部変数と同様な形態で出力コード中に確保される領
域）は必要でない。

【0012】ただし、重要なことは、単一または複数の
入力ファイルで構成される全体のプログラムに対して生
成される目的コード中に最低1個の仮想関数テーブルの
実体が必要なことである。そうしないと、プログラムの
実行中に、オブジェクトは実際に仮想関数テーブルが作
成されていない位置のアドレスを保持することになって
しまう。

【0013】そこで、C++処理系が全てのプログラム
を同時に調べて目的コード中に仮想関数テーブルの実体
を1個だけ生成するという処理を行えばよいが、しか
し全てのプログラムを同時に調べるということは処理時
間、消費メモリ量等の問題から現実的ではない。このた
め、多くのC++処理系では、入力ファイルごとに出力
コードを生成し、後で名前の結合（ある出力コードでは
変数の名前だけを用いて参照し、別の出力コード中にそ
の変数の領域があるという場合に、参照された部分の名
前を実際の領域のアドレスに書き換える処理）を行うと
いう方法をとっている。

【0014】したがって、通常のC++処理系では、各
出力コードにおいてすべて仮想関数テーブルの実体を生
成せざるを得ない。なぜならば、単一の入力ファイルを
対象としている限り、他の入力ファイルに対する出力コ
ードで仮想関数テーブルの実体が生成されているかどう
かはわからないからである。

【0015】

【発明が解決しようとする課題】上述した従来のオブ
ジェクト指向言語処理システムでは、メソッドアドレス格
納テーブルに関しては、実行時に決定されるメソッドの

属するクラスの宣言が入力ファイルに含まれていると、
オブジェクト指向言語処理システムのユーザが特に指定
しない場合は出力コード内にメソッドアドレス格納テー
ブルの実体が必ず生成されていたので、本来、目的コー
ド内に実体がクラス毎に1つ生成されれば十分であるメ
ソッドアドレス格納テーブルがクラスの宣言が取り込ま
れた入力ファイルの数だけ生成され、目的コードのサイ
ズが不必要に大きくなるという問題点があった。

【0016】この問題点に対処するために、オブジェ
クト指向言語処理システムのユーザが、入力ファイルごと
に出力コード内にメソッドアドレス格納テーブルの実体
および宣言を生成するか、あるいはメソッドアドレス格
納テーブルの宣言のみを生成するかを指定し、複数の出
力コード間の参照関係が解決されることにより目的コー
ド内に生成されるメソッドアドレス格納テーブルの実体
の数を減少させるという方法も存在する（AT&T,
“AT&T C++ LANGUAGE SYSTEM
RELEASE 2.1 Release Note
s”, p p. 4-16~4-18）。しかし、クラスの
宣言が入力ファイルに含まれているかどうかを、オブ
ジェクト指向言語処理システムのユーザが判定するのは困
難であるという問題点があった。

【0017】目的コードのサイズを不必要に大きくしな
いためには、メソッドの実行時決定に用いられるメソッ
ドアドレス格納テーブルの数を入力コードの意味が変わ
らない限り少なくすべきである。また、メソッドアドレ
ス格納テーブルの数を減少させる処理は、オブジェクト
指向言語処理システムのユーザの判断によることなく自
動的に行うべきである。

【0018】本発明の目的は、上述の点に鑑み、オブ
ジェクト指向言語処理システムが出力コードへのメソッ
ドアドレス格納テーブルの実体および／または宣言の生成
を制御することにより、目的コード内にメソッドアドレ
ス格納テーブルの実体を1つのみ生成し、目的コードの
大きさを削減するようにしたオブジェクト指向言語処理
システムにおける目的コードサイズ最適化方式を提供す
ることにある。

【0019】

【課題を解決するための手段】本発明のオブジェクト指
向言語処理システムにおける目的コードサイズ最適化方
式は、複数の入力ファイルに対して各々に対応する出力
コードを生成した後に複数の出力コード間の参照関係を
解決して1つの目的コードを生成し、かつオブジェ
クト指向言語のメソッドの実行時決定を目的コード内に静的
に生成されるメソッドアドレス格納テーブルを使用する
ことによって実現するオブジェクト指向言語処理シ
ステムにおいて、クラス宣言の意味解析処理時にメソッドの
所属するクラス名と入力ファイル名とを比較し、一致す
ればメソッドアドレス格納テーブルの実体および宣言を
出力コード内に生成し、一致しなければメソッドアドレ

ス格納テーブルの宣言のみを出力コード内に生成する構文意味解析部を設けたことを特徴とする。

【0020】

【実施例】次に、本発明について図面を参照して詳細に説明する。

【0021】本発明の一実施例として、オブジェクト指向言語で記述されたソースプログラムを入力とし、構文意味解析部においてこれを中間言語形式に変換し、さらにコード生成部でこれを出力コードに変換してファイルに出力する、コンパイル方式のオブジェクト指向言語処理システムの例を挙げる。

【0022】図1は、本発明の一実施例に係るオブジェクト指向言語処理システムにおける目的コードサイズ最適化方式の構成を示すブロック図である。本実施例のオブジェクト指向言語処理システムにおける目的コードサイズ最適化方式は、入力ファイル群11と、構文意味解析部13およびコード生成部14を含むコンパイラ12と、出力コード15と、リンカ16と、目的コード17とから構成されている。

【0023】図2を参照すると、構文意味解析部13の目的コードサイズ最適化処理は、クラス宣言意味解析処理ステップS1と、メソッドアドレス格納テーブル要否判定ステップS2と、入力ファイル名／クラス名比較ステップS3と、メソッドアドレス格納テーブル実体および宣言生成ステップS4と、メソッドアドレス格納テーブル宣言生成ステップS5と、同名入力ファイル存在判定ステップS6と、クラス宣言存在判定ステップS7と、クラス宣言包含ステップS8と、コンパイル済判定ステップS9と、コンパイル未処理設定ステップS10と、同名入力ファイル作成ステップS11と、クラス宣言包含ステップS12と、コンパイル未処理設定ステップS13とからなる。

【0024】次に、このように構成された本実施例のオブジェクト指向言語処理システムにおける目的コードサイズ最適化方式の動作について説明する。

【0025】コンパイラ12は、入力ファイル群11として複数の入力ファイルを受け付ける。コンパイラ12は、入力ファイル群11を構成するすべての入力ファイルについて、1つずつコンパイル処理を行うために構文意味解析部13およびコード生成部14を起動する。

【0026】コンパイラ12において、目的コードサイズ最適化処理は、構文意味解析部13によって行われる。

【0027】最初に、構文意味解析部13は、入力ファイル内のクラス宣言の意味解析処理を行う（ステップS1）。詳しくは、クラス宣言のメンバのシンボルテーブルへの登録等を行う。

【0028】次に、構文意味解析部13は、入力ファイル内のクラス宣言がメソッドアドレス格納テーブルを必要とするかどうかを調べる（ステップS2）。これは、

入力ファイルに記述されているプログラミング言語の仕様に基づき、現在処理対象としているクラスがメソッドの実行時決定を必要とするかどうかにより決定される。C++処理系の場合、virtualの宣言があるか（#includeされる場合を含める）どうかに基づいて決定する。メソッドアドレス格納テーブルを必要としないならば、このクラスに対してこれ以上の処理は行わない。

【0029】現在処理対象としているクラスがメソッドアドレス格納テーブルを必要とする場合、構文意味解析部13は、クラス名と入力ファイル名とが等しいかどうかを調べる（ステップS3）。ただし、入力ファイル名にオブジェクト指向言語処理システム自体またはオブジェクト指向言語処理システムがその上で動作する情報処理システムが定めた付加部分（拡張子）が含まれている場合は、その付加部分を除いた部分をクラス名と比較する。以後、クラス名と入力ファイル名とを比較するときはこの処理を行うものとする。

【0030】クラス名と入力ファイル名とが等しいならば、構文意味解析部13は、出力コード内にメソッドアドレス格納テーブルの実体および宣言を生成して（ステップS4）、このクラスに対してこれ以上の処理は行わない。この結果、メソッドアドレス格納テーブルの実体および宣言は、クラス名と入力ファイル名とが一致する場合のみ出力コード内に生成されることになる。C++処理系の場合、出力コード15中のデータセクションに仮想関数テーブルを生成するとともに、シンボルテーブルにクラスの仮想関数テーブルの属性を生成する（図3参照）。

【0031】クラス名と入力ファイル名とが等しくない場合、構文意味解析部13は、出力コード内にメソッドアドレス格納テーブルの宣言のみを生成して（ステップS5）、クラス名と同名の入力ファイルがコンパイラ12に与えられた入力ファイル群11中に存在するかどうかを調べる（ステップS6）。

【0032】入力ファイル群11中にクラス名と同名の入力ファイルが存在する場合は、構文意味解析部13は、クラス名と同名の入力ファイルの内容を調べ、現在対象としているクラスの宣言が含まれているかどうかを調べる（ステップS7）。クラスの宣言が含まれていれば、このクラスに対してこれ以上の処理を行わない。

【0033】クラスの宣言が含まれていなければ、構文意味解析部13は、クラス名と同名の入力ファイルの内容を変更して現在対象としているクラスの宣言を含める（ステップS8）。C++処理系の場合、シンボルテーブルにクラスの仮想関数テーブルの属性を生成する（図3参照）。さらに、すでにコンパイル処理が終了している場合には（ステップS9でイエス）、コンパイラ12が記録しているコンパイル未処理の入力ファイル群11に含める（ステップS10）。

【0034】ステップS6で、クラス名と同名の入力ファイルがコンパイラ12に与えられた入力ファイル群11の中に存在しない場合は、構文意味解析部13は、新たにクラス名と同名の入力ファイルを作成して（ステップS11）、オブジェクト指向言語処理システムで動作している情報処理システムの適当な位置に置き、この入力ファイル内にクラスの宣言を含め（ステップS12）、さらにこの入力ファイルをコンパイラ12が記録しているコンパイル未処理の入力ファイル群11に含める（ステップS13）。この場合、新たに作成した入力ファイルについてコンパイル処理が施される時点で、出力コード15中にメソッドアドレス格納テーブルの実体および宣言が生成される。

【0035】コンパイラ12により複数の出力コード15が出力された後、リンカ16は、複数の出力コード15間の参照関係を解決して1つの目的コード17を生成する。

【0036】図3は、入力ファイルX.h、X.Cおよびa.Cと出力コードX.oおよびa.oの一例を示す図である。入力ファイルX.hがインクルードされている。クラス名Xと入力ファイル名X.Cの付加部分を除いたものとが一致するので、出力コードX.oにはクラスXの仮想関数テーブルの実体および宣言が生成されている。また、クラス名Xと入力ファイル名a.Cの付加部分を除いたものとが一致しないので、出力コードa.oにはクラスXの仮想関数テーブルの宣言のみが生成されてい

る。

【0037】

【発明の効果】以上説明したように本発明は、オブジェクト指向言語処理システムの構文意味解析部に出力コード内へのメソッドアドレス格納テーブルの実体および／または宣言の生成の選択をオブジェクト指向言語処理システム自体が行う機能を設け、目的コード内にメソッドアドレス格納テーブルの実体を1つのみ生成するようにしたことにより、オブジェクト指向言語処理システムのユーザが困難な解析を行うことなく、目的コードのサイズを削減することができるという効果がある。

【図面の簡単な説明】

【図1】本発明の一実施例に係るオブジェクト指向言語処理システムにおける目的コードサイズ最適化方式の構成を示すブロック図である。

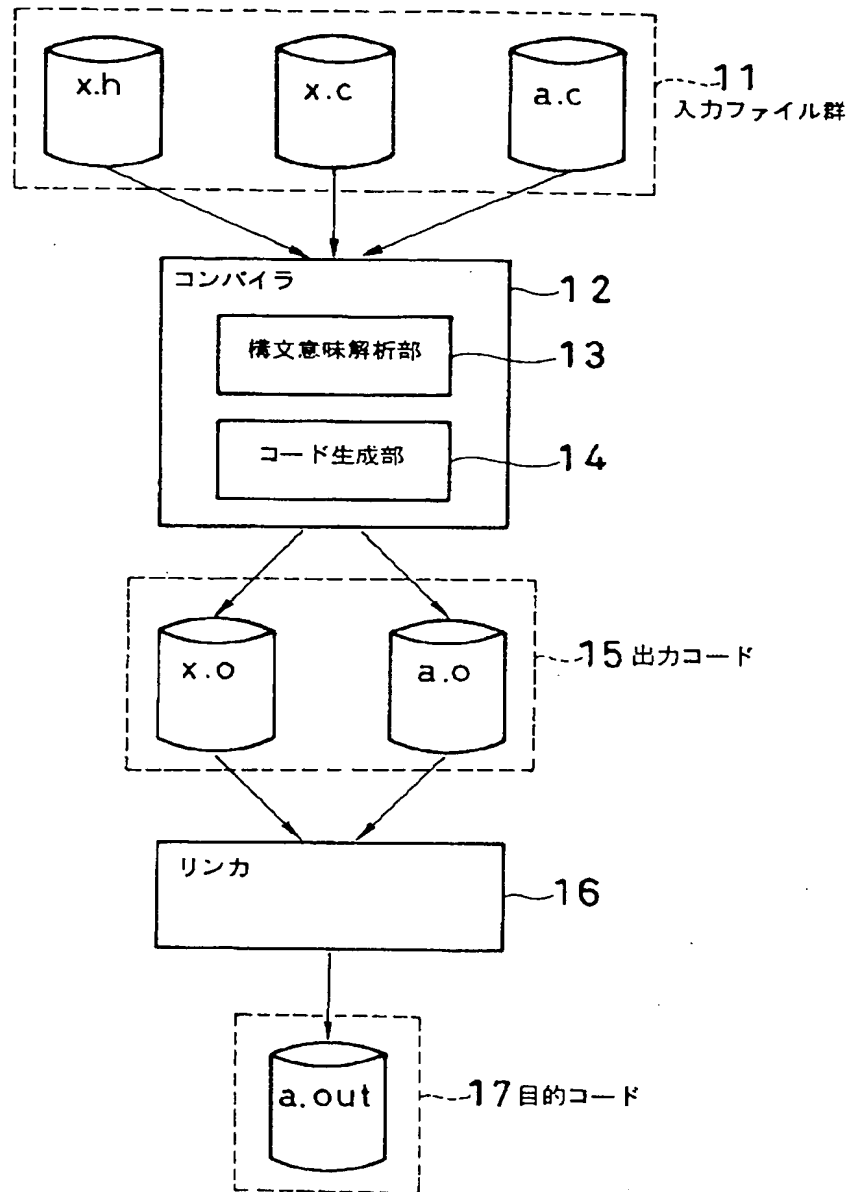
【図2】図1中の構文意味解析部における目的コードサイズ最適化処理を示す流れ図である。

【図3】図1中の入力ファイルおよび出力コードの一例を示す図である。

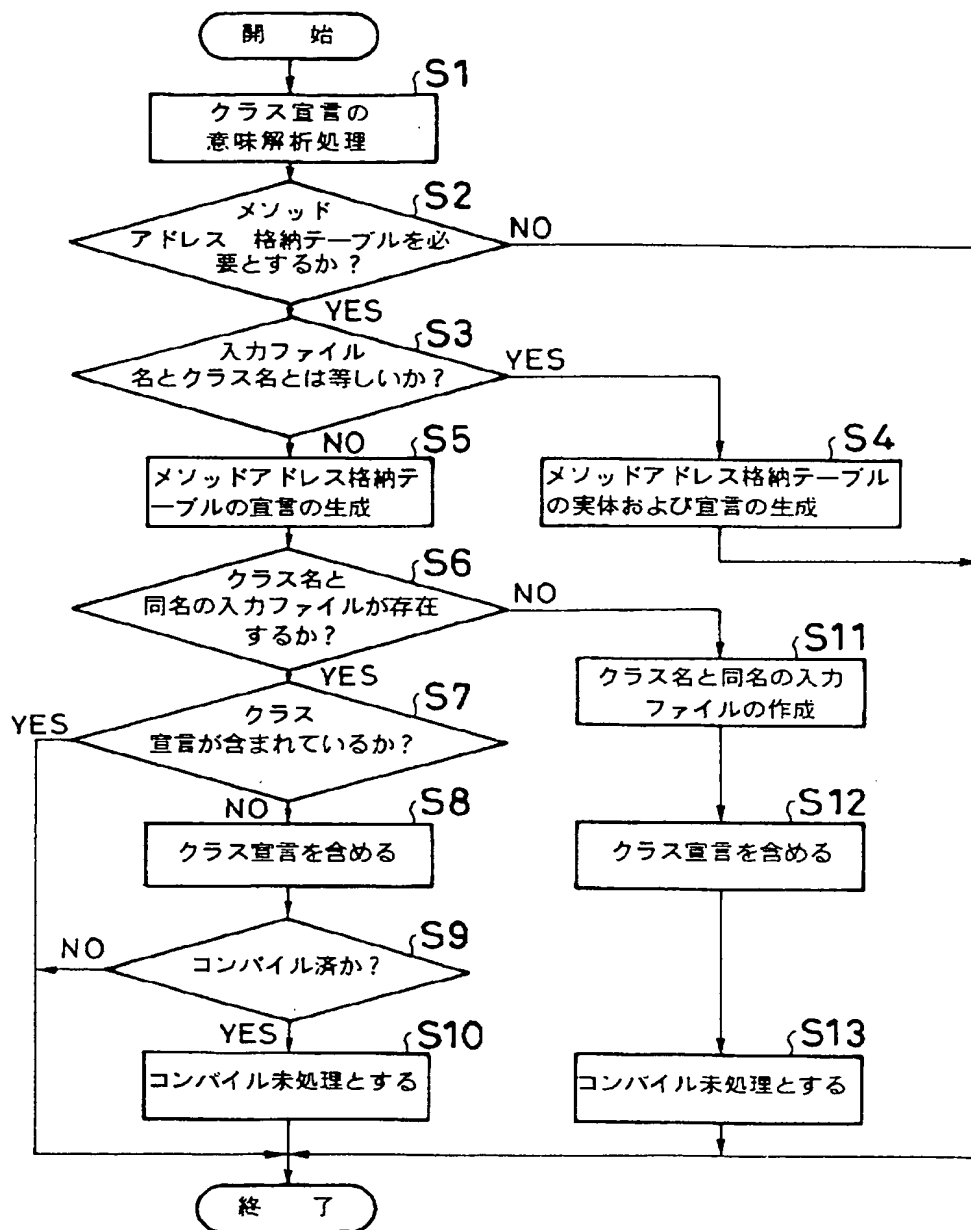
【符号の説明】

- 11 入力ファイル群
- 12 コンパイラ
- 13 構文意味解析部
- 14 コード生成部
- 15 出力コード
- 16 リンカ
- 17 目的コード

【図 1】



【図 2】



【図 3】

